Vers l' intégration d'une approche de génération automatique de mod ele de simulation dans un flot de conception de contrôle -commande

Sophie Prat, Philippe Rauffet, Alain Bignon, Pascal Berruet

Vers l'intégration d'une approche de génération automatique de modèle de simulation dans un flot de conception de contrôle-commande

Sophie PRAT[1,2], Philippe RAUFFET[1], Alain BIGNON[2], Pascal BERRUET[1]

[1]Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance,
UMR 6285 – Université de Bretagne-Sud, CNRS, Centre de Recherche, BP 92116
56321 Lorient Cedex, France.
sophie.prat@univ-ubs.fr ; philippe.rauffet@univ-ubs.fr ; pascal.berruet@univ-ubs.fr

[2]SEGULA Technologies,
BP 50256, 56602 Lanester Cedex, France.
alain.bignon@segula.fr

Résumé— **Dans le cadre d'une démarche outillée d'aide à la conception de contrôle-commande, basée sur l'Ingénierie Dirigée par les Modèles (IDM), on cherche à y intégrer des techniques de vérification par simulation. L'intérêt est de pouvoir permettre la génération automatisée du prototype virtuel du système à commander/superviser, et ce dès le début du cycle de conception. Cette démarche de conception de systèmes sociotechniques permet de générer un programme de commande et l'interface de supervision associée, dès le début de la conception du système. La génération est automatisée grâce à l'utilisation des techniques de l'IDM. En vue de l'obtention d'un prototype virtuel d'un procédé de gestion de fluide, une approche de structuration des modèles de simulation du procédé est présentée dans cet article. Les perspectives d'intégration au sein du flot de conception de l'outil d'aide à la conception *Anaxagore* sont aussi abordées.**

Mots-clés— **Simulation, Vérification, IDM, Aide à la conception.**

I. Introduction

Dans un contexte de concurrence généralisée, l'amélioration de la conception de systèmes sociotechniques (systèmes complexes en forte interaction avec l'humain) doit faire face aux problématiques liées à la communication au sein d'équipes pluridisciplinaires. D'autre part, il est nécessaire de limiter la détection tardive d'erreurs de conception (détectée par exemple, lors de l'implémentation, lorsque le système physique est figé).

C'est dans ce sens que des travaux ont été initiés par Alain Bignon [3], [8], cherchant à générer, dès les premières phases de conception, programmes de commande et interfaces de supervision associées. Grâce à l'utilisation des techniques issues de l'Ingénierie Dirigée par les Modèles (IDM), en particulier les transformations de modèles, la génération est ainsi automatisée. Cependant, à ce jour, l'outil ne dispose pas de techniques de vérifications intégrées qui permettraient de garantir la qualité des résultats générés, tout en y consacrant un minimum de temps. Un intérêt particulier est porté sur la simulation qui permet de rendre compte de la dynamique du comportement du procédé réel, et ainsi d'obtenir un prototype virtuel dès le début de la conception.

Toutefois, l'obtention de modèles de simulation requiert des connaissances approfondies du procédé et en simulation [12], et dépend aussi de l'objectif de la simulation (c'est-à-dire de ce que l'on souhaite vérifier en utilisant la simulation). De plus, avant de pouvoir utiliser la simulation pour effectuer des vérifications sur la commande ou au niveau de l'IHM (Interface Homme-Machine) de supervision, il faut pouvoir en garantir sa qualité. C'est-à-dire répondre à la question ≪est-ce que le modèle de simulation et son exécution sont bien fidèles au comportement du système physique par rapport à l'objectif prédéfini ?≫.

Dans nos travaux nous considérerons le cas de la conception de systèmes de gestion de fluide embarqués sur des navires. Afin de pouvoir vérifier le bon fonctionnement (ou le respect de spécifications), il faudra donc être en mesure de modéliser le comportement de l'évolution du fluide en fonction des commandes implémentées. Du point de vue de l'équipage qui exerce sur le système un pilotage macroscopique, le comportement de celui-ci est discret. Par exemple, pour une pompe de régulation de pression, l'opérateur en charge de la commande du système ne gère pas directement l'asservissement en vitesse de la pompe, mais simplement son état (Marche/Arrêt) ou son mode de fonctionnement (Manuel/Automatique). Les commandes implémentées sur les automates programmables industriels sont également discrètes. Les quelques asservissements continus nécessaires sont généralement implantés dans l'équipement qui, ainsi, s'autorégule. Pour cela, nous retenons l'utilisation d'une simulation à événements discrets (changement d'états à des instants précis dans le temps [13]) pour notre approche.

En vue de sa génération automatique, une approche de structuration des modèles de simulation du procédé est proposée dans cet article. Dans un premier temps l'objectif de la simulation sera définie, puis l'approche de structuration sera présentée et illustrée sur un cas d'étude.

II. Travaux antérieurs et pistes de recherches

L'Ingénierie Dirigée par les modèles (IDM) *renvoie à l'utilisation systématique de modèles comme artefacts de conception primaires dans le cycle de vie. L'IDM est un domaine de l'informatique qui met à disposition des outils, des concepts et des langages pour créer et transfor-*

mer ces modèles [15]. L'IDM est fondée sur la notion de modèle et sur deux concepts principaux. D'une part l'utilisation d'autant de langage de modélisation que nécessaire (concept de *métamodélisation*), et d'autre part le concept de *transformation de modèles* afin de rendre les modèles opérationnels. L'approche MDA (*Model Driven Architecture*) [11] propose de transformer un modèle indépendant de toutes plates-formes d'exécution (PIM) en un modèle spécifique à une plate-forme (PSM). Le PSM obtenu correspond à un modèle opérationnel. Se référer à l'état de l'art réalisé dans [5] pour plus de précisions. Parmi les principaux intérêts identifiés dans [15], on peut retenir la réutilisation de modèles sur étagères, l'interopérabilité, la portabilité, la ré-ingénierie.

Dans divers travaux, l'IDM est utilisée pour faire de la génération automatique. Parmi les différentes applications on peut retenir : l'obtention de programmes de commande, d'IHM, ou encore de modèles de simulation. Certaines approches permettent également de faire de la génération conjointe.

Dans ce cadre, des travaux ont été menés pour créer un outil d'aide à la conception générant conjointement programme de commande et interface de supervision. Cet outil [3], [8], nommé *Anaxagore*, utilise les techniques de transformations de modèles de l'IDM pour générer la commande et l'interface de supervision à partir d'un modèle métier (fourni par un mécanicien) et d'une bibliothèque d'éléments. Le modèle métier est un schéma P&ID (*Piping & Instrumentation Diagram*) décrivant le système physique. Un élément est défini comme l'unité constitutive du procédé du système. Il peut être relatif à du matériel (vanne, pompe ...) ou à des fonctionnalités du système. Chaque élément de la bibliothèque contient plusieurs *vues*. Le concept de *vue* est ici une extension de celui utilisé par Lallican [9], il correspond aux différents points de vues des concepteurs.

D'autres travaux concernent la génération conjointe de code de commande et de modèles de simulation. On peut citer à ce sujet les travaux de Lallican [9] sur les systèmes transitiques, puis étendus par Bévan [2] aux systèmes transitiques reconfigurables. La génération se fait à partir d'une description du système, exprimée dans un langage spécifique au domaine (DSL) et d'une bibliothèque de composants (au sens de Lallican). Les vues *parties opératives* des composants sont utilisées pour obtenir, par transformation de modèle, le modèle de simulation de la partie opérative du système transitique. Ce dernier est exécutable sur le simulateur SimSED. A noter que la simulation conjointe de la commande et de la partie opérative est utilisée pour vérifier le code de commande généré. Adam, dans ses travaux [1], génère un observateur basé sur un simulateur à événement discret. Cet observateur a pour but de fournir, au système d'Aide à la Décision, l'état du système de production. A partir d'un même modèle du système, est généré conjointement la commande et l'observateur. Dans le cadre de la génération automatique de modèles de simulation, il est fait état que la génération passe généralement par l'utilisation d'une bibliothèque de modèles de simulation *élémentaires* [1] qui sont ensuite instanciés et paramétrés. L'avantage est que la génération s'en trouve facilitée et cela permet une capitalisation de la connaissance, et la réutilisation. De plus, les données d'entrées sont le plus souvent sous forme de modèles UML transformés ensuite en modèle de simulation. Dans les travaux de El Haouzi [6], l'implémentation d'une plate-forme de simulation générique est faite à partir d'un modèle de connaissance en UML. Une instanciation de la plate-forme générique permet d'obtenir la plate-forme de simulation pour un système particulier. Le logiciel ARENA est utilisé pour la simulation. On peut aussi avoir des modèles SysML. Comme par exemple dans [7] où le diagramme d'activité est transformé en Réseau de Petri puis transformé en VHDL-AMS pour être exécuté sur le logiciel System Vision. Dans le domaine de l'électronique, on peut citer les travaux de Sarmento [16] où il est généré le modèle de simulation global d'un système hétérogène embarqué afin de valider le système. Pour ce faire la co-simulation est utilisée.

De façon générale, on distingue dans les méthodes de modélisation pour la simulation basée sur l'IDM, deux approches : l'une basée sur un langage unifié, l'autre sur un langage spécifique au domaine [10]. Cependant, l'utilisation d'un modèle UML (ou SysML) ne semble pas pertinent dans le cadre de l'outil *Anaxagore*. Cela demanderait aux experts de développer un nouveau modèle du système. Il semble plus judicieux de réutiliser les modèles métiers réalisés pour la conception du système, sans rajouter une charge de travail supplémentaire aux concepteurs. De plus, comme dans les travaux de Bévan et d'Adam, la bibliothèque utilisée est basée sur le concept de *vue*. De ce fait, il apparaît envisageable de rajouter à la génération conjointe de la commande et de l'IHM de supervision, celles des modèles de simulation.

L'intérêt de notre démarche (*Fig.1*) est d'enrichir le flot déjà existant de l'outil *Anaxagore* en y intégrant la génération des modèles de simulation. La simulation permettra d'effectuer des vérifications au plus tôt. C'est-à-dire que l'on cherche à utiliser la simulation non seulement sur les modèles de sorties, mais aussi sur leurs modèles intermédiaires.

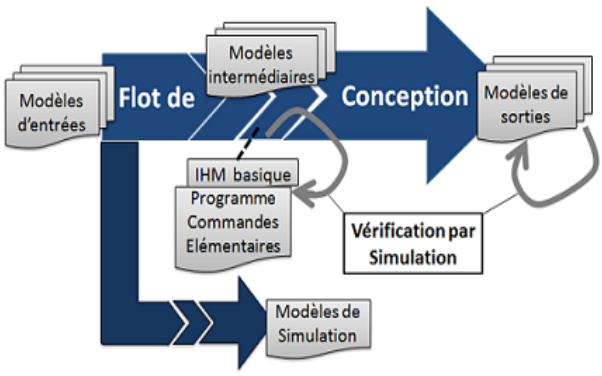

Fig. 1. Place de la simulation dans le flot de conception

Contrairement aux systèmes de production, les systèmes de gestion de fluide embarqués nécessitent de modéliser non seulement le comportement de la partie opérative, mais aussi celle du fluide. En vue d'obtenir des représentations génériques, il semble judicieux de dissocier ces deux modélisations, notamment par rapport à la réutilisation des modèles [16]. En effet, le comportement d'un élément de la partie opérative n'est pas dépendant du fluide. Par contre

les effets de la partie opérative sur le fluide sont liés à sa nature. Comme il convient de simuler à la fois le comportement de la partie opérative et celle du fluide, il faudra structurer les modèles de simulation associés à la partie opérative et au fluide.

III. Approche retenue en vue d'une génération automatique de modèles de simulation

L'intérêt de simuler le comportement du procédé est de pouvoir effectuer des vérifications dynamiques. Les différentes interactions entre l'IHM de supervision, la partie de contrôle/commande et la simulation sont schématisées sur la *Fig.2*.

Des vérifications peuvent concerner l'aide à la décision et la surveillance. Par exemple, on peut chercher à évaluer au niveau de l'IHM, si les informations disponibles sont nécessaires et suffisantes, ou bien s'intéresser au système d'alarmes (au niveau du contrôle/commande ou de l'IHM). En effet, des alarmes peuvent être distinctes lorsqu'elles sont transmises à l'IHM de supervision, mais ne pas être affichées comme telle sur l'IHM. De plus, il ne faut pas que des alarmes contradictoires puissent être actives en même temps. D'autres vérifications peuvent concerner l'adaptation du système face aux aléas, ou encore s'intéresser aux commandes de haut niveau.

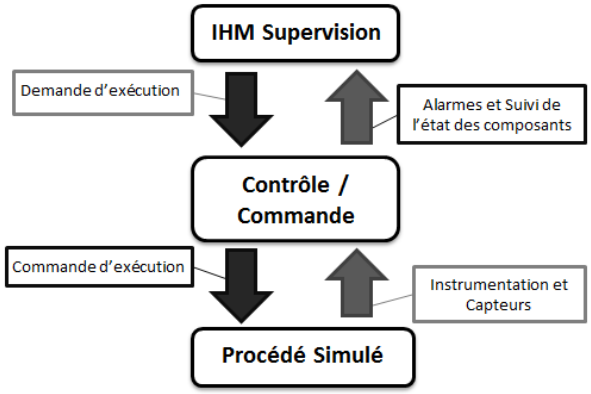

Fig. 2. Schéma des interactions

Cette vérification d'exigences fonctionnelles nécessite de modéliser le comportement de la partie opérative d'une part, et l'évolution du comportement du fluide d'autre part. Cependant, notre approche est utilisée en tout début de conception (le système physique n'est donc pas encore figé). Il en résulte un manque considérable de données quantitatives, ne permettant pas de modéliser finement le comportement du fluide. Par exemple, les dimensions réelles des canalisations sont inconnues. De ce fait, il n'apparaît pas pertinent de s'intéresser à des exigences quantitatives telle que la qualité de service. L'objectif de la simulation est donc de vérifier qualitativement des exigences fonctionnelles sur des procédés de gestion de fluide.

A. Structuration des modèles de Simulation

On propose de structurer les modèles de simulation en trois niveaux (*Fig.3*). Le Niveau *Composant* correspond à la modélisation du comportement des éléments commandés (partie opérative) et de l'instrumentation. Le Niveau *Contextualisation* correspond à la contextualisation des effets des opérations sur le fluide. Enfin, un Niveau *Système*, contient les règles d'évolution de l'environnement.

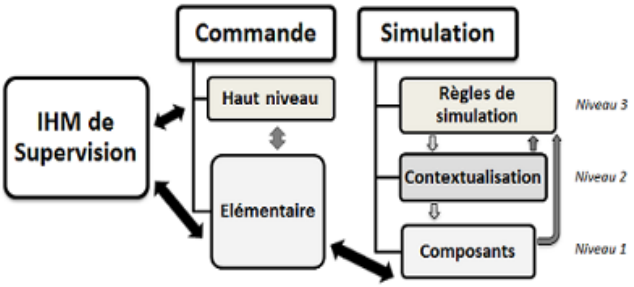

Fig. 3. Schéma des interactions entre IHM, Commande et Simulation du procédé

Les échanges d'informations entre partie commande et partie simulation se font entre le Niveau *Élémentaire* de la partie commande et le Niveau *Composant* de la partie simulation.

Les échanges d'informations internes à la partie simulation sont multiples. En effet le Niveau *système*, en fonction de l'état de la partie opérative (Niveau *Composant*) et du fluide (Niveau *contextualisation*), fait une mise à jour des informations du Niveau *Contextualisation*. Ces informations concernent essentiellement les règles de propagation de l'évolution du fluide. Le Niveau *Contextualisation*, quant à lui, renvoie des informations destinées à l'instrumentation au Niveau *Composant*. Enfin, le Niveau *Composant* retourne les informations sur les capteurs et les mesures à la partie commande.

B. Illustration sur un cas d'étude

Le cas d'étude retenu a été simplifié ici pour des raisons de place, mais reste néanmoins représentatif des effets existant sur un système auxiliaire global.

Le système présenté sur la *Fig.4* est constitué de deux soutes (*St1* et *St2*) séparées par une vanne motorisée à deux voies (*VM1*). La soute *St2* est reliée à une vanne motorisée (*VM2*). Chaque vanne est constituée d'un capteur de fin de course d'ouverture (`FcO`) et de fermeture (`FcF`).

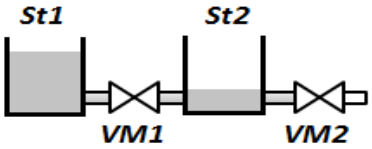

Fig. 4. Schéma du cas d'étude

L'illustration se fera en utilisant les modèles de simulation exécutables, exprimés avec des langages de programmation de la norme IEC 61131-3. Ainsi les Niveaux *Composant* et *Contextualisation* sont réalisés en langage FBD (Function Block Diagram), le Niveau *Système* quant à lui est en langage ST (Structured Text).

Le Niveau 1 (*Fig.5*), dit de *Composant*, correspond à la modélisation du comportement des deux vannes (*VM1* et *VM2*). Les entrées `Def_FcO` et `Def_FcF` du bloc fonctionnel `Simu_VM` permettent l'introduction de défauts sur les capteurs de fin de course. Les commandes d'ouverture et de fermeture sont `CmdO` et `CmdF`. Les sorties `Ouverte` et `Fermee` correspondent à la position de la vanne (ouverte

ou fermée). Enfin `FcO` et `FcF` correspondent aux valeurs des capteurs de fin de course qui sont envoyées à la partie commande. Lorsqu'il n'y a pas d'introduction de défauts, `FcO` (respectivement `FcF`) a la même valeur que `Ouverte` (respectivement `Fermee`).

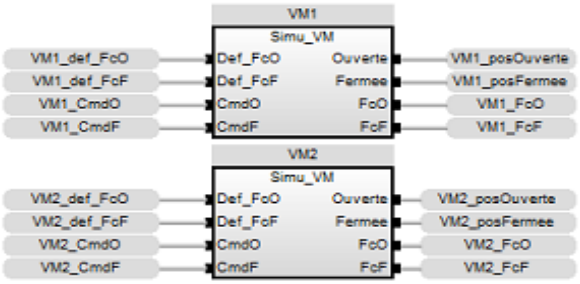

Fig. 5. Implémentation du Niveau *composant* en FBD

L'évolution des changements de positions de la vanne (et par conséquent, l'évolution des valeurs des variables `Ouverte` et `Fermee`) peut être modélisée par Réseau de Petri (*Fig.6*).

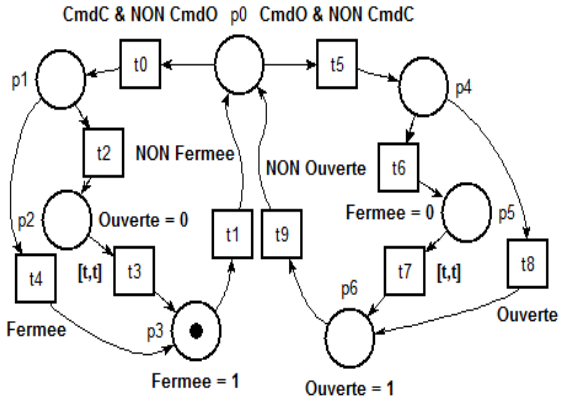

Fig. 6. RdP modélisant le comportement de la vanne

L'hypothèse a été faite que la prise en compte d'une commande d'ouverture ou fermeture se fait uniquement lorsque la vanne est en position fermée ou ouverte. Le marquage du Réseau de Petri sur la *Fig.6*, signifie que la vanne est dans la position fermée. La transition `t1` est sensibilisée dès lors que la place `p3` est marquée. La condition associée au franchissement de cette transition est donc toujours vraie (il en va de même pour la transition `t9`). Une fois cette transition franchie, la place `p0` devient marquée. La vanne est alors en attente d'une commande d'ouverture ou fermeture. Lors d'une commande de fermeture, la transition `t0` est franchie. La place `p1` reçoit le jeton de la place `p0`. La vanne étant déjà en position fermée, les transitions `t4` puis `t1` sont franchies. La place `p0` est à nouveau marquée. Lors d'une commande d'ouverture, la transition `t5` est franchie. La vanne n'étant pas déjà ouverte, la transition `t6` est franchie. La place `p5` devient alors marquée. La vanne commençant sa phase d'ouverture, elle quitte sa position fermée. A la fin du temps de manoeuvre, elle se retrouve en position ouverte, la place `p6` devient ainsi marquée.

Le Niveau 2 (*Fig.8*), dit de *Contextualisation*, présente la modélisation des effets de l'état des vannes sur le fluide, ainsi que la modélisation du fluide (ici de l'eau) dans les soutes. En d'autres termes, on modélise le fait qu'une vanne laisse passer ou non le débit de fluide (bloc fonctionnel `Eau_VM`), et l'évolution du niveau de fluide dans les soutes (bloc fonctionnel `Eau_St`).

En s'inspirant de la modélisation sous forme d'activité des travaux de [4], on peut modéliser les effets sur le fluide (*Fig.7*). Par exemple, une vanne pourrait avoir deux activités : *laissez passer le fluide* et *empêcher le fluide de passer*.

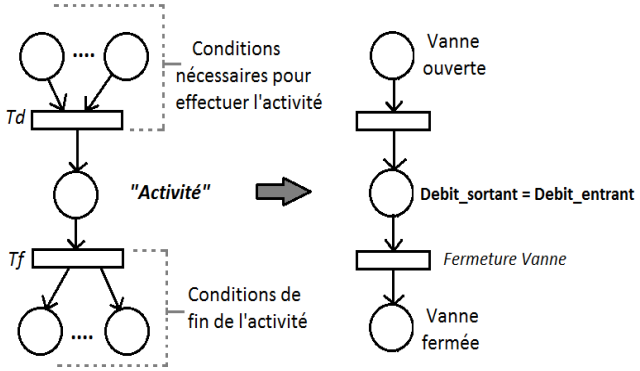

Fig. 7. Modélisation sous forme d'activité, application pour l'activité *laisser passer le fluide* d'une vanne

Pour qu'une *activité* débute, il faut que toutes les conditions nécessaires soient remplies. Par conséquent, une fois que toutes les places correspondant à ces conditions sont marquées, la transition `Td` est franchie, l'activité commence. La transition `Tf` sera franchie, lorsque les conditions pour effectuer l'activité ne sont plus remplies, ce sera donc la fin de l'activité.

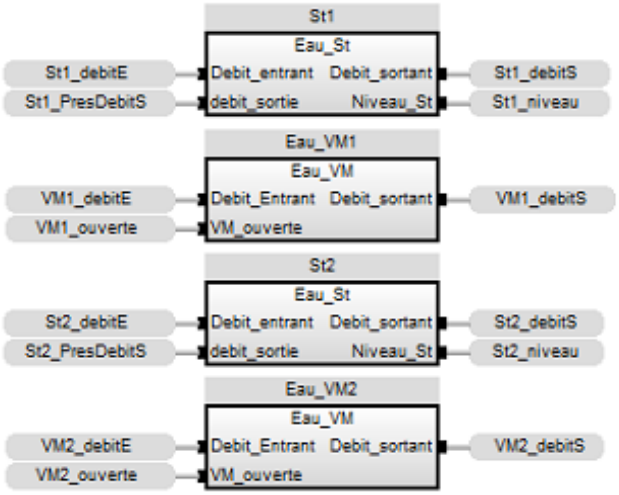

Fig. 8. Implémentation du Niveau2 de simulation en FBD

Les entrées du bloc fonctionnel `Eau_VM` sont : `Debit_Entrant` et `VM_ouverte`. Le premier correspond au débit de fluide entrant dans la vanne, et le second au fait que la vanne est ouverte. La sortie du bloc fonctionnel, `Debit_Sortant`, correspond au débit sortant de la vanne. Ainsi lorsque `VM_ouverte = TRUE`, le débit en sortie de la vanne est le même que celui en entrée (si l'on néglige la perte de charge). Au contraire, si `VM_ouverte = FALSE` alors le débit sortant de la vanne

sera nul (`Debit_sortant = 0`). Les entrées du bloc fonctionnel `Eau_St` sont : `Debit_entrant` et `debit_sortie`. Le premier correspond au débit de fluide arrivant dans la soute, et le second au fait qu'il y ait un débit sortant de la soute ou non. En effet, la présence ou non d'un débit sortant de la soute est lié, en plus du niveau de fluide stocké, à l'architecture du système. Les sorties du bloc fonctionnel sont `Debit_sortant` et `Niveau_St`. Le premier correspond au débit qui sort de la soute, et le second au niveau de fluide dans la soute. Ainsi la soute se remplira (augmentation du niveau du fluide) lorsqu'il y a uniquement un débit en entrée de la soute (c'est-à-dire que `debit_sortie` aura pour valeur `FALSE` et `Debit_entrant` aura une valeur non nulle). La soute se videra lorsqu'il y a uniquement un débit sortant de la soute (et que le niveau de fluide est suffisant). C'est-à-dire quand `Debit_entrant` aura une valeur nulle et `debit_sortie` aura la valeur `TRUE`. Dans le cas où la soute serait remplie et vidée en même temps (brassage d'une soute), alors `Debit_entrant` aura une valeur non nulle et `debit_sortie` aura la valeur `TRUE`. Le niveau dans la soute sera constant si le débit de remplissage est le même que celui de vidage.

Le Niveau 3 (*Fig.9*), dit *système*, représente les règles de propagations du comportement du fluide. En effet, c'est à ce niveau que les liens entre les différents blocs fonctionnels du Niveau *contextualisation* seront effectués. Dans cet exemple, les échanges d'informations entre les différents niveaux de simulation, sont réalisés à l'aide de variables globales.

```
VM1_ouverte := NOT VM1_posFermee;
VM2_ouverte := NOT VM2_posFermee;        Règle 1

If VM1_ouverte Then
   St1_PresDebitS := (St1_niveau > St2_niveau);
Else
   St1_PresDebitS := FALSE;              Règle 2
End_If;

VM1_debitE := St1_debitS;
St2_debitE := VM1_debitS;                Règle 4

If VM2_ouverte Then
   St2_PresDebitS := (St2_niveau > 0);
Else
   St2_PresDebitS := FALSE;              Règle 3
End_If;

VM2_debitE := St2_debitS;                Règle 4
```

Fig. 9. Exemple d'implémentation du Niveau *système* en ST

On considère dans cet exemple, qu'une vanne est ouverte, dès lors qu'elle n'est pas en position fermée. Ceci donnera lieu à une première règle de simulation.

Règle 1 :
Affectation des valeurs d'entrée du paramètre `VM_ouverte` des instances des blocs fonctionnels `Eau_VM`.

Ainsi, il est nécessaire de connaître et de récupérer les informations concernant la position d'ouverture des vannes (obtenues dans le Niveau *Composant*). Elles sont disponibles à travers les variables `VM1_posOuverte` (pour la vanne *VM1*) et *VM2_posOuverte* (pour `VM2`). L'affectation des valeurs d'entrée du paramètre `VM_ouverte` (dans le Niveau *Contextualisation*) se fait à l'aide des variables `VM1_ouverte` et `VM2_ouverte`, qui prennent ici la valeur des variables `VM1_posOuverte` et `VM2_posOuverte`.

Au vu de l'architecture du procédé, il va falloir prendre en compte le phénomène des vases communicants (le niveau dans les soutes tendant naturellement à s'équilibrer). Ainsi la présence ou non d'un débit sortant de la soute *St1* est non seulement liée à l'état de la vanne *VM1* mais aussi au niveau d'eau dans les deux soutes. En effet, même si la vanne *VM1* est ouverte, lorsque la soute *St2* se sera remplie au même niveau que la soute *St1* alors le débit sortant de celle-ci sera nul. Ce qui donne lieu à une deuxième règle de simulation.

Règle 2 :
Affectation de la valeur d'entrée de `debit_sortie` de l'instance *St1* du bloc fonctionnel `Eau_St`.

Ainsi, `debit_sortie` vaut `True` lorsque la vanne *VM1* est ouverte (d'après la règle 1) et que le niveau dans la soute (`Niveau_St`) est supérieur à celui de la soute *St2*. Dans le cas contraire `debit_sortie` vaut `False`. L'information concernant le niveau dans les soutes est fait par l'intermédiaire des variables `St1_niveau` et `St2_niveau`. L'affectation de la valeur d'entrée `debit_sortie` de l'instance *St1* (Niveau *Contextualisation*), se fait par la variable `St1_PresDebitS`. Concernant la soute *St2*, la présence ou non d'un débit sortant de la soute est directement liée au fait que la vanne *VM2* soit ouverte ou non, et qu'il reste ou non de l'eau dans cette soute. On aura donc une troisième règle de simulation.

Règle 3 :
Affectation de la valeur d'entrée de `debit_sortie` de l'instance *St2* du bloc fonctionnel `Eau_St`. `debit_sortie` vaut `True` lorsque la vanne *VM2* est ouverte (d'après la règle 1), et que le niveau dans la soute (`Niveau_St`) est non nul.

Enfin, il reste à connecter les débits sortant et entrant des différentes instances des blocs fonctionnels `Eau_VM` et `Eau_St` pour modéliser l'écoulement du fluide.

Règle 4 :
Propagation des débits entre les différents composants.

Ainsi, le débit sortant de la soute *St1* (variable `St1_debitS`) étant relié au débit entrant dans la vanne *VM1*, la variable *VM1_debitE* sera affectée de la valeur de `St1_debitS`. Le débit sortant de cette même vanne est relié au débit entrant dans la soute *St2*. La variable `St2_debitE` sera donc affectée par la valeur de `VM1_debitS`. Enfin, le débit sortant de la soute *St2* étant relié au débit entrant dans la vanne *VM2*, `VM1_debitE` sera donc affectée par la valeur de `St2_debitS`.

Les interactions entre les différents niveaux de simulation sont illustrées en partie sur la *Fig.10*. On y voit la récupération de l'état du procédé (partie opérative et du fluide) par le Niveau *Système*, qui après évaluation des règles de simulation, met à jour les variables du niveau *Contextualisation*. Pour plus de clarté, seule la soute *St1* et la vanne *VM1* ont été représentées. Pour la même raison, l'ordre potentiel d'évaluation des règles de simulation, et la synchronisation des mises à jour des variables d'entrée d'un même bloc n'ont pas été représentés. De même, l'instrumentation ne figurant pas dans le cas d'étude, la transmission des informations du Niveau *Contextualisation* destinée à l'instrumentation n'est pas représentée.

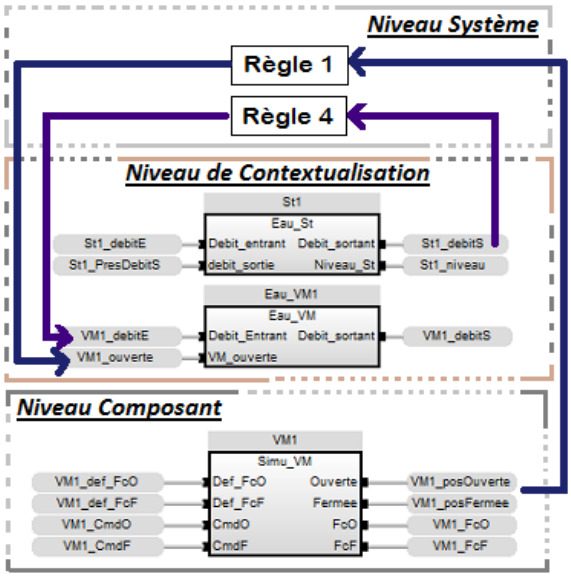

Fig. 10. Illustration des interactions entre les niveaux de simulation

IV. Discussion : Perspective d'intégration dans le flot de conception de l'outil Anaxagore

En reprenant le concept de bibliothèque de simulation, la bibliothèque d'éléments standards de l'outil *Anaxagore* pourrait être enrichie par une *vue* de simulation. Cette *vue* de simulation contiendrait les modèles des Niveau *Composant* et *Contextualisation* (Niveau 1 et 2). Cela permettrait ainsi de pouvoir générer, pour ces niveaux, les modèles de simulation du procédé. Un paramétrage sera cependant nécessaire. Le Niveau 3 paraît quant-à-lui, pour l'instant, difficilement entièrement automatisable en l'état. En effet, les données (ou informations) nécessaires ne sont que très peu formalisées.

Au vu du principal objectif de l'outil cherchant à limiter le temps passé à faire des spécifications, il n'est pas forcément judicieux de chercher à formaliser toute la connaissance du système que possèdent les experts. Une alternative, pour faciliter l'implémentation du Niveau 3, consisterait à fournir une bibliothèque de règles de simulation. L'expert choisirait les règles de simulation nécessaire pour le procédé à simuler, qu'il paramétrerait ensuite, ainsi que leur ordre d'évaluation.

Toutefois certaines règles sont automatisables, comme celle concernant la propagation des débits. Les informations nécessaires sont déjà formalisées. Elles sont d'ailleurs utilisées pour générer les animations de propagation du fluide sur l'IHM de supervision.

V. Conclusion

Cette proposition de structuration apporte une méthode de construction des modèles de simulation, en vue de la génération automatique d'un prototype virtuel de procédé de gestion de fluide. La dissociation de la partie opérative et du comportement de fluide permet une modélisation modulaire, avec différents niveaux d'abstraction possibles. Ainsi, en fonction du besoin, on peut affiner la finesse de reproduction du comportement du fluide, sans pour autant devoir modifier celle de la partie opérative (et réciproquement). La modélisation sous forme de RdP semble être une piste intéressante pour introduire la validation et la vérification des modèles de simulation.

L'utilisation de la simulation, en fin du flot de conception, devrait permettre d'effectuer les vérifications sur le système complet et de valider l'interface de supervision auprès du client. En allant plus loin, cela permettrait de réaliser des tests utilisateurs ou encore de former les futurs utilisateurs du système.

Une mise en pratique de cette approche est en cours de réalisation, sur le même principe que le cas d'étude présenté, afin de réaliser un prototypage rapide d'un système de *production, stockage, et distribution d'eau douce* embarqué sur un navire.

Références

[1] Adam M., Cardin O., Berruet P., et Castagna P. Proposal of an Approach to Automate the Generation of a Transitic System's Observer and Decision Support using Model Driven Engineering. IFAC World Congress, Milan, Italie, 28 Août au 2 Septembre 2011.

[2] Bévan R. Approche composant pour la commande multi-versions des systèmes transitiques reconfigurables. Thèse de doctorat de l'Université de Bretagne-Sud, 9 Décembre 2013.

[3] Bignon A., Rossi A. et Berruet P. An integrated design flow for the joint generation of control and interfaces from a business model. *Computer in Industry*, vol. 64, n° 6, pp 634–649, 2013.

[4] Combacau M., Courvoisier M. Process failures diagnosis in FMS real-time control : an approach combining rule-based systems and Petri nets. The Second Annual Conference on AI, Simulation and Planing in High Autonomy Systems, Cocoa Beach, FL, Etat-Unis. 1–2 Avril 1991.

[5] Combemale B. Ingénierie Dirigée par les Modèles (IDM) – État de l'art. 2008.

[6] El Haouzi H., Pannequin R., Thomas A. Génération automatique de plateformes de simulation pour des systèmes organisés en flux tirés. 7e Congrès International de Génie Industriel, Trois Rivières, Canada, 5–8 juin 2007.

[7] Foures D., Albert V., Pascal J-C. et Nketsa A. Automation of SysML activity diagram simulation with model-driven engineering approach. Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium, Orlando, FL, Etat-Unis, 26–29 Mars 2012.

[8] Goubali O., Bignon A., Berruet P., Girard P. et Guittet L. Anaxagore, an exemple of model-driven engineering for industrial supervision. Ergonomie et Informatique Avancée Conference, Ergo'IA, Fance, 15–17 octobre 2014.

[9] Lallican J. L. Proposition d'une approche composant pour la conception de la commande des systèmes transitiques. Thèse de doctorat de l'Université de Bretagne-Sud, 12 Décembre 2007.

[10] Li X., Lei Y., Wang W., Wang W. et Zhu Y. A DSM-based multi-paradigm simulation modeling approach for complex systems. 2013 Winter Simulation Conference, Washington, DC., Etat-Unis, 8–11 Décembre 2013.

[11] Kennedy A, Carter K., Frank W., et Architects D. MDA Guide Version 1.0, 2003.

[12] Pritsker A.A.B., Henriksen J.O, Fishwick P.A. et Clark G.M. Principles of modeling. 1991 Winter Simulation Conference, Phoenix, AZ, Etat-Unis, 8–11 Décembre 1991.

[13] Ray C. ATLAS, une plate-forme pour la modélisation et la simulation de système désagrégés. Thèse de doctorat de l'Université de Rennes I, 19 Septembre 2003.

[14] Robinson S., Nance R.E., Paul R.J., Pidd M. et Taylor S.J.E. Simulation model reuse : definitions, benefits and obstacles. *Simulation Model Practice Theory*, vol. 12, n°7–8, pp.479–494, 2004.

[15] Rochet S. Formalisation des Processus de l'Ingénierie Système : Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application. Thèse de doctorat de l'Université de Paul Sabatier – Toulouse III, 26 Novembre 2007

[16] Sarmento A. Génération Automatique de Modèles de Simulation pour la Validation de Systèmes Hétérogènes Embarqué. Thèse de doctorat de l'Université Joseph Fournier – Grenoble I, 28 Octobre 2005.